My favorites ▼ | Sign in
Google Analytics
Google+ Platform

re2

an efficient, principled regular expression library

Search projects

Syntax

I define UNIX as "30 definitions of regular expressions living under one roof." —Don Knuth

Updated Jan 9, 2014 by rsc@swtch.com

This page lists the regular expression syntax accepted by RE2.
It also lists syntax accepted by PCRE, PERL, and VIM.
Grayed out expressions are not supported by RE2.

Single characters:

.	any character, possibly including newline (s=true)
[xyz]	character class
[^xyz]	negated character class
\d	Perl character class
\D	negated Perl character class
[:alpha:]	ASCII character class
[:^alpha:]	negated ASCII character class
\pN	Unicode character class (one-letter name)
\p{Greek}	Unicode character class
\PN	negated Unicode character class (one-letter name)
\P{Greek}	negated Unicode character class

Composites:

xy	x followed by y
x|y	x or y (prefer x)

Repetitions:

x*	zero or more x, prefer more
x+	one or more x, prefer more
x?	zero or one x, prefer one
x{n,m}	n or n+1 or ... or m x, prefer more
x{n,}	n or more x, prefer more
x{n}	exactly n x
x*?	zero or more x, prefer fewer
x+?	one or more x, prefer fewer
x??	zero or one x, prefer zero
x{n,m}?	n or n+1 or ... or m x, prefer fewer
x{n,}?	n or more x, prefer fewer
x{n}?	exactly n x
x{}	(≡ x*) (NOT SUPPORTED) VIM
x{-}	(≡ x*?) (NOT SUPPORTED) VIM
x{-n}	(≡ x{n}?) (NOT SUPPORTED) VIM
x=	(≡ x?) (NOT SUPPORTED) VIM

Possessive repetitions:

x*+	zero or more x, possessive (NOT SUPPORTED)
x++	one or more x, possessive (NOT SUPPORTED)
x?+	zero or one x, possessive (NOT SUPPORTED)
x{n,m}+	n or ... or m x, possessive (NOT SUPPORTED)
x{n,}+	n or more x, possessive (NOT SUPPORTED)
x{n}+	exactly n x, possessive (NOT SUPPORTED)

Grouping:

(re)	numbered capturing group
(?P<name>re)	named & numbered capturing group
(?<name>re)	named & numbered capturing group (NOT SUPPORTED)
(?'name're)	named & numbered capturing group (NOT SUPPORTED)
(?:re)	non-capturing group
(?flags)	set flags within current group; non-capturing

(?flags:re)	set flags during re; non-capturing
(?#text)	comment (NOT SUPPORTED)
(?|x|y|z)	branch numbering reset (NOT SUPPORTED)
(?>re)	possessive match of re (NOT SUPPORTED)
re@>	possessive match of re (NOT SUPPORTED) VIM
%(re)	non-capturing group (NOT SUPPORTED) VIM

Flags:

i	case-insensitive (default false)
m	multi-line mode: ^ and $ match begin/end line in addition to begin/end text (default false)
s	let . match \n (default false)
U	ungreedy: swap meaning of x* and x*?, x+ and x+?, etc (default false)

Flag syntax is xyz (set) or -xyz (clear) or xy-z (set xy, clear z).

Empty strings:

^	at beginning of text or line (m=true)
$	at end of text (like \z not \Z) or line (m=true)
\A	at beginning of text
\b	at word boundary (\w on one side and \W, \A, or \z on the other)
\B	not a word boundary
\G	at beginning of subtext being searched (NOT SUPPORTED) PCRE
\G	at end of last match (NOT SUPPORTED) PERL
\Z	at end of text, or before newline at end of text (NOT SUPPORTED)
\z	at end of text
(?=re)	before text matching re (NOT SUPPORTED)
(?!re)	before text not matching re (NOT SUPPORTED)
(?<=re)	after text matching re (NOT SUPPORTED)
(?<!re)	after text not matching re (NOT SUPPORTED)
re&	before text matching re (NOT SUPPORTED) VIM
re@=	before text matching re (NOT SUPPORTED) VIM
re@!	before text not matching re (NOT SUPPORTED) VIM
re@<=	after text matching re (NOT SUPPORTED) VIM
re@<!	after text not matching re (NOT SUPPORTED) VIM
\zs	sets start of match (= \K) (NOT SUPPORTED) VIM
\ze	sets end of match (NOT SUPPORTED) VIM
\%^	beginning of file (NOT SUPPORTED) VIM
\%$	end of file (NOT SUPPORTED) VIM
\%V	on screen (NOT SUPPORTED) VIM
\%#	cursor position (NOT SUPPORTED) VIM
\%'m	mark m position (NOT SUPPORTED) VIM
\%23l	in line 23 (NOT SUPPORTED) VIM
\%23c	in column 23 (NOT SUPPORTED) VIM
\%23v	in virtual column 23 (NOT SUPPORTED) VIM

Escape sequences:

\a	bell (≡ \007)
\f	form feed (≡ \014)
\t	horizontal tab (≡ \011)
\n	newline (≡ \012)
\r	carriage return (≡ \015)
\v	vertical tab character (≡ \013)
*	literal *, for any punctuation character *
\123	octal character code (up to three digits)
\x7F	hex character code (exactly two digits)
\x{10FFFF}	hex character code
\C	match a single byte even in UTF-8 mode
\Q...\E	literal text ... even if ... has punctuation
\1	backreference (NOT SUPPORTED)
\b	backspace (NOT SUPPORTED) (use \010)
\cK	control char ^K (NOT SUPPORTED) (use \001 etc)
\e	escape (NOT SUPPORTED) (use \033)
\g1	backreference (NOT SUPPORTED)
\g{1}	backreference (NOT SUPPORTED)
\g{+1}	backreference (NOT SUPPORTED)

\g{-1}	backreference (NOT SUPPORTED)
\g{name}	named backreference (NOT SUPPORTED)
\g<name>	subroutine call (NOT SUPPORTED)
\g'name'	subroutine call (NOT SUPPORTED)
\k<name>	named backreference (NOT SUPPORTED)
\k'name'	named backreference (NOT SUPPORTED)
\lX	lowercase X (NOT SUPPORTED)
\ux	uppercase x (NOT SUPPORTED)
\L...\E	lowercase text ... (NOT SUPPORTED)
\K	reset beginning of $0 (NOT SUPPORTED)
\N{name}	named Unicode character (NOT SUPPORTED)
\R	line break (NOT SUPPORTED)
\U...\E	upper case text ... (NOT SUPPORTED)
\X	extended Unicode sequence (NOT SUPPORTED)
\%d123	decimal character 123 (NOT SUPPORTED) VIM
\%xFF	hex character FF (NOT SUPPORTED) VIM
\%o123	octal character 123 (NOT SUPPORTED) VIM
\%u1234	Unicode character 0x1234 (NOT SUPPORTED) VIM
\%U12345678	Unicode character 0x12345678 (NOT SUPPORTED) VIM

Character class elements:

x	single character
A-Z	character range (inclusive)
\d	Perl character class
[:foo:]	ASCII character class foo
\p{Foo}	Unicode character class Foo
\pF	Unicode character class F (one-letter name)

Named character classes as character class elements:

[\d]	digits (≡ \d)
[^\d]	not digits (≡ \D)
[\D]	not digits (≡ \D)
[^\D]	not not digits (≡ \d)
[[:name:]]	named ASCII class inside character class (≡ [:name:])
[^[:name:]]	named ASCII class inside negated character class (≡ [:^name:])
[\p{Name}]	named Unicode property inside character class (≡ \p{Name})
[^\p{Name}]	named Unicode property inside negated character class (≡ \P{Name})

Perl character classes:

\d	digits (≡ [0-9])
\D	not digits (≡ [^0-9])
\s	whitespace (≡ [\t\n\f\r])
\S	not whitespace (≡ [^\t\n\f\r])
\w	word characters (≡ [0-9A-Za-z_])
\W	not word characters (≡ [^0-9A-Za-z_])
\h	horizontal space (NOT SUPPORTED)
\H	not horizontal space (NOT SUPPORTED)
\v	vertical space (NOT SUPPORTED)
\V	not vertical space (NOT SUPPORTED)

ASCII character classes:

[:alnum:]	alphanumeric (≡ [0-9A-Za-z])
[:alpha:]	alphabetic (≡ [A-Za-z])
[:ascii:]	ASCII (≡ [\x00-\x7F])
[:blank:]	blank (≡ [\t])
[:cntrl:]	control (≡ [\x00-\x1F\x7F])
[:digit:]	digits (≡ [0-9])
[:graph:]	graphical (≡ [!-~] == [A-Za-z0-9!"#$%&'()*+,\-./:;<=>?@[\\\]^_`{|}~])
[:lower:]	lower case (≡ [a-z])
[:print:]	printable (≡ [-~] == [[:graph:]])
[:punct:]	punctuation (≡ [!-/:-@[-`{-~])
[:space:]	whitespace (≡ [\t\n\v\f\r])
[:upper:]	upper case (≡ [A-Z])
[:word:]	word characters (≡ [0-9A-Za-z_])

[:xdigit:]	hex digit (≡ [0-9A-Fa-f])

Unicode character class names--general category:

C	other
Cc	control
Cf	format
Cn	unassigned code points (NOT SUPPORTED)
Co	private use
Cs	surrogate
L	letter
LC	cased letter (NOT SUPPORTED)
L&	cased letter (NOT SUPPORTED)
Ll	lowercase letter
Lm	modifier letter
Lo	other letter
Lt	titlecase letter
Lu	uppercase letter
M	mark
Mc	spacing mark
Me	enclosing mark
Mn	non-spacing mark
N	number
Nd	decimal number
Nl	letter number
No	other number
P	punctuation
Pc	connector punctuation
Pd	dash punctuation
Pe	close punctuation
Pf	final punctuation
Pi	initial punctuation
Po	other punctuation
Ps	open punctuation
S	symbol
Sc	currency symbol
Sk	modifier symbol
Sm	math symbol
So	other symbol
Z	separator
Zl	line separator
Zp	paragraph separator
Zs	space separator

Unicode character class names--scripts:

Arabic	Arabic
Armenian	Armenian
Balinese	Balinese
Bamum	Bamum
Batak	Batak
Bengali	Bengali
Bopomofo	Bopomofo
Brahmi	Brahmi
Braille	Braille
Buginese	Buginese
Buhid	Buhid
Canadian_Aboriginal	Canadian Aboriginal
Carian	Carian
Chakma	Chakma
Cham	Cham
Cherokee	Cherokee
Common	characters not specific to one script
Coptic	Coptic
Cuneiform	Cuneiform

Cypriot	Cypriot
Cyrillic	Cyrillic
Deseret	Deseret
Devanagari	Devanagari
Egyptian_Hieroglyphs	Egyptian Hieroglyphs
Ethiopic	Ethiopic
Georgian	Georgian
Glagolitic	Glagolitic
Gothic	Gothic
Greek	Greek
Gujarati	Gujarati
Gurmukhi	Gurmukhi
Han	Han
Hangul	Hangul
Hanunoo	Hanunoo
Hebrew	Hebrew
Hiragana	Hiragana
Imperial_Aramaic	Imperial Aramaic
Inherited	inherit script from previous character
Inscriptional_Pahlavi	Inscriptional Pahlavi
Inscriptional_Parthian	Inscriptional Parthian
Javanese	Javanese
Kaithi	Kaithi
Kannada	Kannada
Katakana	Katakana
Kayah_Li	Kayah Li
Kharoshthi	Kharoshthi
Khmer	Khmer
Lao	Lao
Latin	Latin
Lepcha	Lepcha
Limbu	Limbu
Linear_B	Linear B
Lycian	Lycian
Lydian	Lydian
Malayalam	Malayalam
Mandaic	Mandaic
Meetei_Mayek	Meetei Mayek
Meroitic_Cursive	Meroitic Cursive
Meroitic_Hieroglyphs	Meroitic Hieroglyphs
Miao	Miao
Mongolian	Mongolian
Myanmar	Myanmar
New_Tai_Lue	New Tai Lue (aka Simplified Tai Lue)
Nko	Nko
Ogham	Ogham
Ol_Chiki	Ol Chiki
Old_Italic	Old Italic
Old_Persian	Old Persian
Old_South_Arabian	Old South Arabian
Old_Turkic	Old Turkic
Oriya	Oriya
Osmanya	Osmanya
Phags_Pa	'Phags Pa
Phoenician	Phoenician
Rejang	Rejang
Runic	Runic
Saurashtra	Saurashtra
Sharada	Sharada
Shavian	Shavian
Sinhala	Sinhala
Sora_Sompeng	Sora Sompeng

Sundanese	Sundanese
Syloti_Nagri	Syloti Nagri
Syriac	Syriac
Tagalog	Tagalog
Tagbanwa	Tagbanwa
Tai_Le	Tai Le
Tai_Tham	Tai Tham
Tai_Viet	Tai Viet
Takri	Takri
Tamil	Tamil
Telugu	Telugu
Thaana	Thaana
Thai	Thai
Tibetan	Tibetan
Tifinagh	Tifinagh
Ugaritic	Ugaritic
Vai	Vai
Yi	Yi

Vim character classes:

\i	identifier character (NOT SUPPORTED) VIM
\I	\i except digits (NOT SUPPORTED) VIM
\k	keyword character (NOT SUPPORTED) VIM
\K	\k except digits (NOT SUPPORTED) VIM
\f	file name character (NOT SUPPORTED) VIM
\F	\f except digits (NOT SUPPORTED) VIM
\p	printable character (NOT SUPPORTED) VIM
\P	\p except digits (NOT SUPPORTED) VIM
\s	whitespace character (≡ [\t]) (NOT SUPPORTED) VIM
\S	non-white space character (≡ [^ \t]) (NOT SUPPORTED) VIM
\d	digits (≡ [0-9]) VIM
\D	not \d VIM
\x	hex digits (≡ [0-9A-Fa-f]) (NOT SUPPORTED) VIM
\X	not \x (NOT SUPPORTED) VIM
\o	octal digits (≡ [0-7]) (NOT SUPPORTED) VIM
\O	not \o (NOT SUPPORTED) VIM
\w	word character VIM
\W	not \w VIM
\h	head of word character (NOT SUPPORTED) VIM
\H	not \h (NOT SUPPORTED) VIM
\a	alphabetic (NOT SUPPORTED) VIM
\A	not \a (NOT SUPPORTED) VIM
\l	lowercase (NOT SUPPORTED) VIM
\L	not lowercase (NOT SUPPORTED) VIM
\u	uppercase (NOT SUPPORTED) VIM
\U	not uppercase (NOT SUPPORTED) VIM
_x	\x plus newline, for any x (NOT SUPPORTED) VIM

Vim flags:

\c	ignore case (NOT SUPPORTED) VIM
\C	match case (NOT SUPPORTED) VIM
\m	magic (NOT SUPPORTED) VIM
\M	nomagic (NOT SUPPORTED) VIM
\v	verymagic (NOT SUPPORTED) VIM
\V	verynomagic (NOT SUPPORTED) VIM
\Z	ignore differences in Unicode combining characters (NOT SUPPORTED) VIM

Magic:

(?{code})	arbitrary Perl code (NOT SUPPORTED) PERL
(??{code})	postponed arbitrary Perl code (NOT SUPPORTED) PERL
(?n)	recursive call to regexp capturing group n (NOT SUPPORTED)
(?+n)	recursive call to relative group +n (NOT SUPPORTED)
(?-n)	recursive call to relative group -n (NOT SUPPORTED)
(?C)	PCRE callout (NOT SUPPORTED) PCRE

(?R)	recursive call to entire regexp (≡ (?0)) (NOT SUPPORTED)
(?&name)	recursive call to named group (NOT SUPPORTED)
(?P=name)	named backreference (NOT SUPPORTED)
(?P>name)	recursive call to named group (NOT SUPPORTED)
(?(cond)true|false)	conditional branch (NOT SUPPORTED)
(?(cond)true)	conditional branch (NOT SUPPORTED)
(*ACCEPT)	make regexps more like Prolog (NOT SUPPORTED)
(*COMMIT)	(NOT SUPPORTED)
(*F)	(NOT SUPPORTED)
(*FAIL)	(NOT SUPPORTED)
(*MARK)	(NOT SUPPORTED)
(*PRUNE)	(NOT SUPPORTED)
(*SKIP)	(NOT SUPPORTED)
(*THEN)	(NOT SUPPORTED)
(*ANY)	set newline convention (NOT SUPPORTED)
(*ANYCRLF)	(NOT SUPPORTED)
(*CR)	(NOT SUPPORTED)
(*CRLF)	(NOT SUPPORTED)
(*LF)	(NOT SUPPORTED)
(*BSR_ANYCRLF)	set \R convention (NOT SUPPORTED) PCRE
(*BSR_UNICODE)	(NOT SUPPORTED) PCRE

Comment by davide.s...@gmail.com, Aug 12, 2011

WTF?

Comment by lennym...@gmail.com, Nov 18, 2011

So the major trade offs are back-references for example: (abc)\1 and not matching look-behinds. In exchange you get high speed regex. There are wrappers for many popular languages too. This can be very useful for certain applications.

Comment by david.e...@gmail.com, Jan 12, 2012

For definitions of various unicode classes for \p{Class} syntax, the following data files are useful:

- http://unicode.org/Public/UNIDATA/PropertyValueAliases.txt
- http://www.unicode.org/Public/UNIDATA/UnicodeData.txt

Comment by sysco...@gmail.com, Feb 21, 2013

(*UCP) is missing from this list. In PCRE, this sets the PCRE_UCP option, which adds unicode property support, and changes \w to match alphanumeric from unicode properties.

Comment by Localsys...@gmail.com, Nov 10, 2013

<r>

Comment by jvar...@georgiasouthern.edu, Nov 13, 2013

If I try [\d]{3} for entering three digits the expression does not match. I guess I am reading this wrongly...

Comment by faithsm...@gmail.com, Dec 10, 2013

ok,it's a good way to learn golang

Comment by jkto...@google.com, Dec 31, 2013

jvar..., try using \d{3}.

Comment by justas...@gmail.com, Jan 28, 2014

"><img src=x onerror=confirm(1);>

Comment by d...@google.com, Mar 21, 2014

"Grayed out expressions are not supported by RE2" would be fine if the gray could be readily discerned from black. Please use a color (like red) or *italic* or ~~strikeout~~, or actually use a light gray background (or even better, a ~~*combination*~~).

Comment by awbloc...@google.com, Apr 29, 2014

+1 for strikeout.

Comment by raghav.e...@gmail.com, Jun 18, 2014

+1 for color or strikeout, its painful to differentiate between supported and non-supported.

Comment by robotic1...@googlemail.com, Jun 19, 2014

+1 for color (#ff0000)

Comment by sarun...@gmail.com, Jun 27 (4 days ago)

For those who want to change color of (NOT SUPPORTED) items in this page. Just enter this into developer console:

```
(function(){
var allTr=document.getElementsByTagName("tr");
for(var i=0; i<allTr.length; i++)
if((allTr[i].innerHTML.indexOf("NOT SUPPORT")>=0) &&
!(allTr[i].innerHTML.indexOf("<tr")>=0)){
allTr[i].style.color="red";
}
})();
```

► Sign in to add a comment

Terms - Privacy - Project Hosting Help

Powered by Google Project Hosting